PROCEDURES FOR USING TESTING TOOL

Release Date: November 27, 2000

Contract Number: GS-35F-4919H
Order Number: T0000AJ3739

CLIN #012-1

Prepared for:

U. S. DEPARTMENT OF EDUCATION

OFFICE OF STUDENT FINANCIAL ASSISTANCE (SFA)
400 Maryland Avenue, SW
Washington, DC 20202

Prepared by:

CTGi
Suite 250
10461 White Granite Drive
Oakton, VA 22124

This page left intentionally left blank

FOREWORD

CTG, Incorporated (CTGi) would like to thank the following personnel whose dedication and involvement made the development and completion of this document possible.

Kelly Conboy

Candace Jones

Frank Majewski

Annette Mazie

Antony Mosley

Isaac Sanvee

Paul Stocks

Sandra Stocks
Director of Operations, Washington
CTGi

This page intentionally left blank

EXECUTIVE SUMMARY

This document, prepared for the United States (U.S.) Department of Education, Office of Student Financial Assistance (SFA), develops a standardized procedure for using an automated test tool in software test execution and management.

This document, along with those listed below, will be integrated into the U.S. Department of Education, SFA, System Integration and Testing (SI&T) Process Handbook, which will then become integrated into the overall U.S. Department of Education, SFA, Modernization Technology Handbook. The remaining documents that will comprise the U.S. Department of Education, SFA, SI&T Process Handbook are:

- System Integration and Testing Standards
- Test Performance Measurements
- Procedures and Templates for Creating Test Conditions, Test Scenarios, and Testing Data
- Procedures and Templates for Test Execution, Test Evaluation, and Error Correction
- Procedures and Templates for System Configuration Management (CM) and Quality Assurance (QA)

Each of the above listed procedures, templates, and guides were prepared and delivered as a separate document.

All SI&T guidelines and procedures are focused on supporting systems or projects used in the development and execution of a comprehensive integration and testing program. To this end, this document contains information on understanding issues related to using automated testing tools, responsibilities of the organization, and required testing practices.

This page intentionally left blank

TABLE OF CONTENTS

Section **Page**

1. INTRODUCTION **1-1**
1.1 Background 1-1
1.2 Objective 1-1
1.3 Applicability 1-1
1.4 Document Organization 1-1

2. PROCEDURES FOR USING TESTING TOOL **2-1**
2.1 Overview of Using a Testing Tool 2-1
2.2 Configuring the Test Tool 2-3
2.2.1 Participants 2-3
2.2.2 Entrance Criteria 2-4
2.2.3 Inputs 2-4
2.2.4 Activities 2-4
2.2.5 Outputs 2-5
2.2.6 Exit Criteria 2-5
2.3 Defining Software Requirements 2-5
2.3.1 Participants 2-5
2.3.2 Entrance Criteria 2-6
2.3.3 Inputs 2-6
2.3.4 Activities 2-6
2.3.5 Outputs 2-7
2.3.6 Exit Criteria 2-7
2.4 Defining Test Cases And Test Requirements 2-7
2.4.1 Participants 2-7
2.4.2 Entrance Criteria 2-8
2.4.3 Inputs 2-8
2.4.4 Activities 2-8
2.4.5 Outputs 2-9
2.4.6 Exit Criteria 2-9
2.5 Procedures For Creating Automated Test Scripts 2-9
2.5.1 Participants 2-9
2.5.2 Entrance Criteria 2-10
2.5.3 Inputs 2-10
2.5.4 Activities 2-10
2.5.5 Outputs 2-10
2.5.6 Exit Criteria 2-11
2.6 Procedures For Executing Automated Test Scripts 2-11
2.6.1 Participants 2-11
2.6.2 Entrance Criteria 2-11
2.6.3 Inputs 2-12
2.6.4 Activities 2-12
2.6.5 Outputs 2-12
2.6.6 Exit Criteria 2-12

TABLE OF CONTENTS (Cont'd)

Section **Page**

2.7 Procedures for Tracking System Problem Reports 2-13
2.7.1 Participants 2-17
2.7.2 Entrance Criteria 2-17
2.7.3 Inputs 2-18
2.7.4 Activities 2-18
2.7.5 Outputs 2-18
2.7.6 Exit Criteria 2-18

GLOSSARY **Gls-1**

BIBLIOGRAPHY **Bbl-1**

APPENDIX A TESTING TOOL PROCEDURES CHECK-OFF LIST **A-1**

APPENDIX B SPR TRACKING DATABASE TABLE DEFINITIONS **B-1**

APPENDIX C SOFTWARE REQUIREMENT TRACKING TABLE LAYOUT **C-1**

LIST OF EXHIBITS

Figure 2-1. Procedures For Using An Automated Test Tool 2-3
Figure 2.2 System Problem Report Tracking 2-15

This page intentionally left blank

LIST OF ACRONYMS

CM	Configuration Management
CSCI	Computer Software Configuration Item
DCR	Document Change Request
HWCI	Hardware Configuration Item
QA	Quality Assurance
SD	System Development
SDF	Software Development File
SRD	Software Requirements Document
SU	Software Unit
SFA	Student Financial Assistance
SI&T	System Integration and Testing
SPR	System Problem Report
SQT	System Qualification Testing
STD	System Test Description
STP	System Test Plan
STR	System Test Report

This page intentionally left blank

1. INTRODUCTION

1.1 Background

The U. S. Department of Education, Office of Student Financial Assistance (SFA), contracted CTG, Incorporated (CTGi), in August 2000, to develop standardized System Integration and Test (SI&T) procedures. These procedures will be used for guidance, planning, and implementation involving current and future U. S. Department of Education, SFA, enterprise information technology systems projects.

1.2 Objective

This objective of this document is to provide the procedures necessary for standardization when using an automated test tool to support and implement all U.S. Department of Education, SFA, information technology SI&T projects.

Creating standardized procedures for using a testing tool is critical to establishing a consistent and comprehensive test capability. These standard procedures will be documented and available to all personnel involved in the oversight and testing of software applications. The purpose of these procedures is to promote a consistent approach in both development and testing of application products. Standard integration and testing processes include essentials on what is to be done, why it is to be done, and who will perform and complete the work. Procedural activities are provided to explain how to complete the process (the sequence of tasks or task steps to be performed), when the work is performed and completed, and the criteria for measuring quality of the work.

1.3 Applicability

When the SI&T process is performed by either the U. S. Department of Education, SFA, staff and/or contractors, this document applies, unless specifically excluded, in the program/project plan, contract, etc. This document is used for the creation of guidelines and procedures for the planning, preparation, execution, analysis, and evaluation, of all types of U. S. Department of Education, SFA, information technology project integration and testing.

1.4 Document Organization

This document contains two narrative sections, a Glossary, a Bibliography, and three appendices. Section 1, Introduction, provides brief background information and states the guiding objective and applicability for the document. Section 2, Procedures for Using Test Tools, provides the basis and procedures for using a test tool in support of a software project testing effort. Appendix A, Testing Tool Procedures Check-Off List, provides a checklist that serves as a guide when using automated test procedures. Appendix B, SPR Tracking Database Table Definitions, provides a table of data definitions needed for a System Problem Report (SPR) tracking database. Appendix C, Software Requirement Tracking Table Layout, provides tables of software, test, and traceability requirements needed to implement a requirements tracking database.

This page intentionally left blank

2. PROCEDURES FOR USING TESTING TOOL

The use of a comprehensive set of testing tools promotes consistent and efficient software test management. The term "test tool" is used in this document to describe a collection of databases and software products that assist in the test effort. The test effort includes planning for testing, performing the testing, and reporting the results of testing. Procedural activities are provided to use a test tool to support all test efforts.

2.1 Overview of Using a Testing Tool

The standardized processes and procedures described in this document are components of SI&T. The purpose of SI&T is to plan and monitor the system testing effort and control testing resources. Testing tools assist in planning, monitoring, and controlling efforts.

SI&T processes involve planning and monitoring system testing efforts at various life-cycle phases or testing levels. The Software Unit (SU) Test phase testing is planned and accomplished by the developers.

Formal testing is planned during Software Requirements Document (SRD) analysis and is accomplished by independent testers. Integration Test phase testing involves testing combinations of Computer Software Configuration Items (CSCI) and their interfaces. Integration Test phase testing is planned during the test design phase and is accomplished with an appropriate balance of developers, with design knowledge, and independent testers.

A significant part of Integration Test phase testing is regression testing. Regression testing is specifically important during maintenance and when upgrading software applications. Regression testing is required to ensure that software changes made to a production version software application do not adversely affect the functionality of the original software application.

Performance Test phase testing involves testing the entire system to verify that certain requirements are met. These requirements include specified function, quality, and performance characteristics. The System Qualification Testing (SQT) phase involves functionally testing the system to ensure that it is ready for delivery. SQT is accomplished, or witnessed, by intended system users.

The purpose of all formal testing is to detect system discrepancies. When a discrepancy is encountered a System Problem Report (SPR) is submitted to record information regarding the detected problems. Collecting information on the types of SPRs filed and the phase at which the SPR was discovered monitors the effectiveness of the various levels of testing.

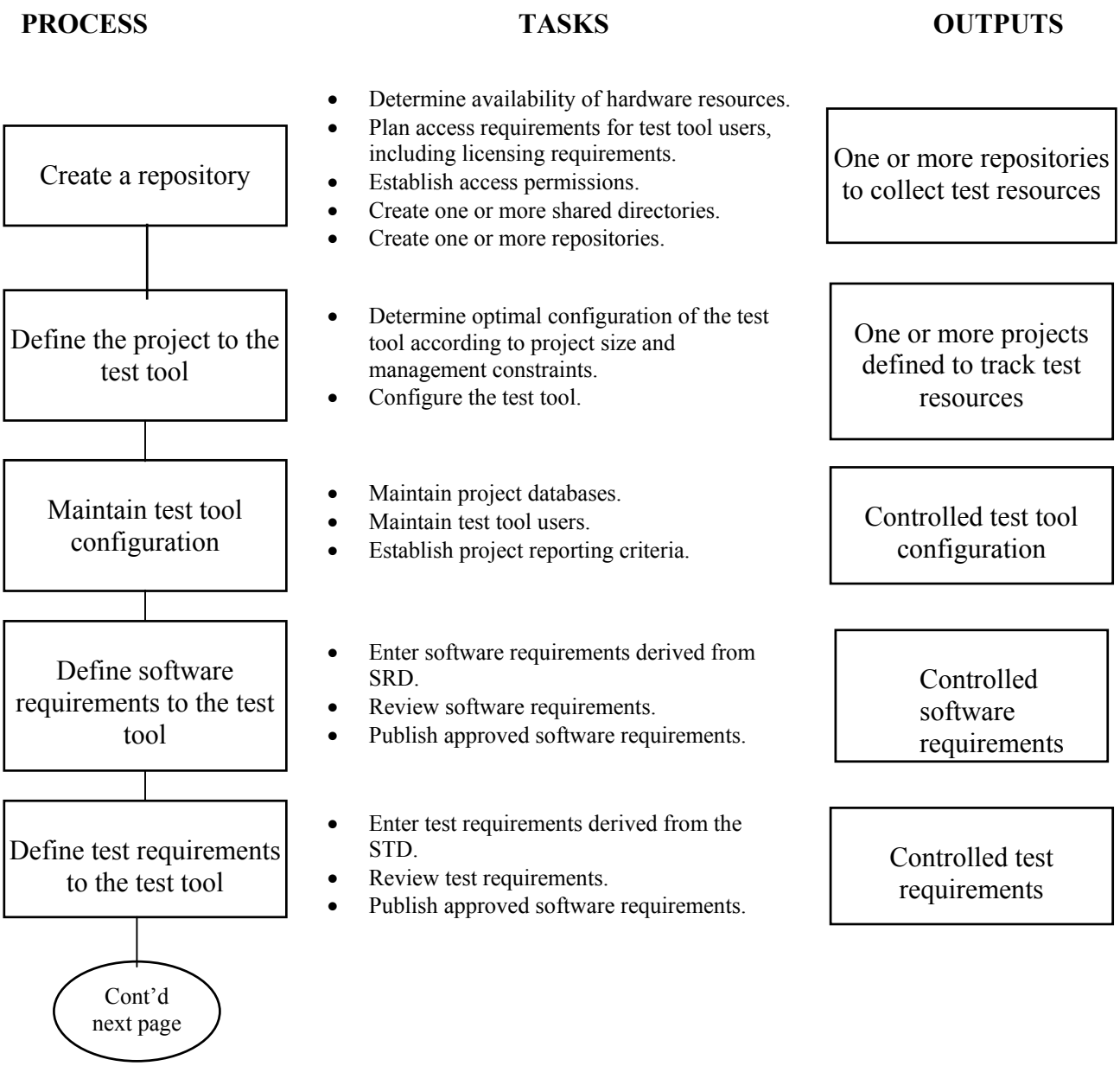
Testing coverage is investigated to discern the amount of code exercised and the number of requirements tested. Efforts to maximize testing efforts and minimize input efforts, by automated means, are investigated and implemented.

The following test resources are tracked and controlled by the test tool:

- Software requirements defined by the SRD.
- Test requirements defined by the System Test Description (STD).
- Automated test script(s), as defined by the STD.
- SPRs, as determined at each testing level.

These test resources are aids that may be drawn upon, as needed. At the end of the SI&T process, the test resources are turned over to the personnel or group(s) responsible for operation and maintenance of the software. The test resources are re-usable for the remainder of the system life cycle.

The source of each test resource and the set of procedures that comprise the SI&T process are discussed, in detail, in the remainder of this document. Figure 2.1 depicts the procedures for using a test tool to support SI&T.



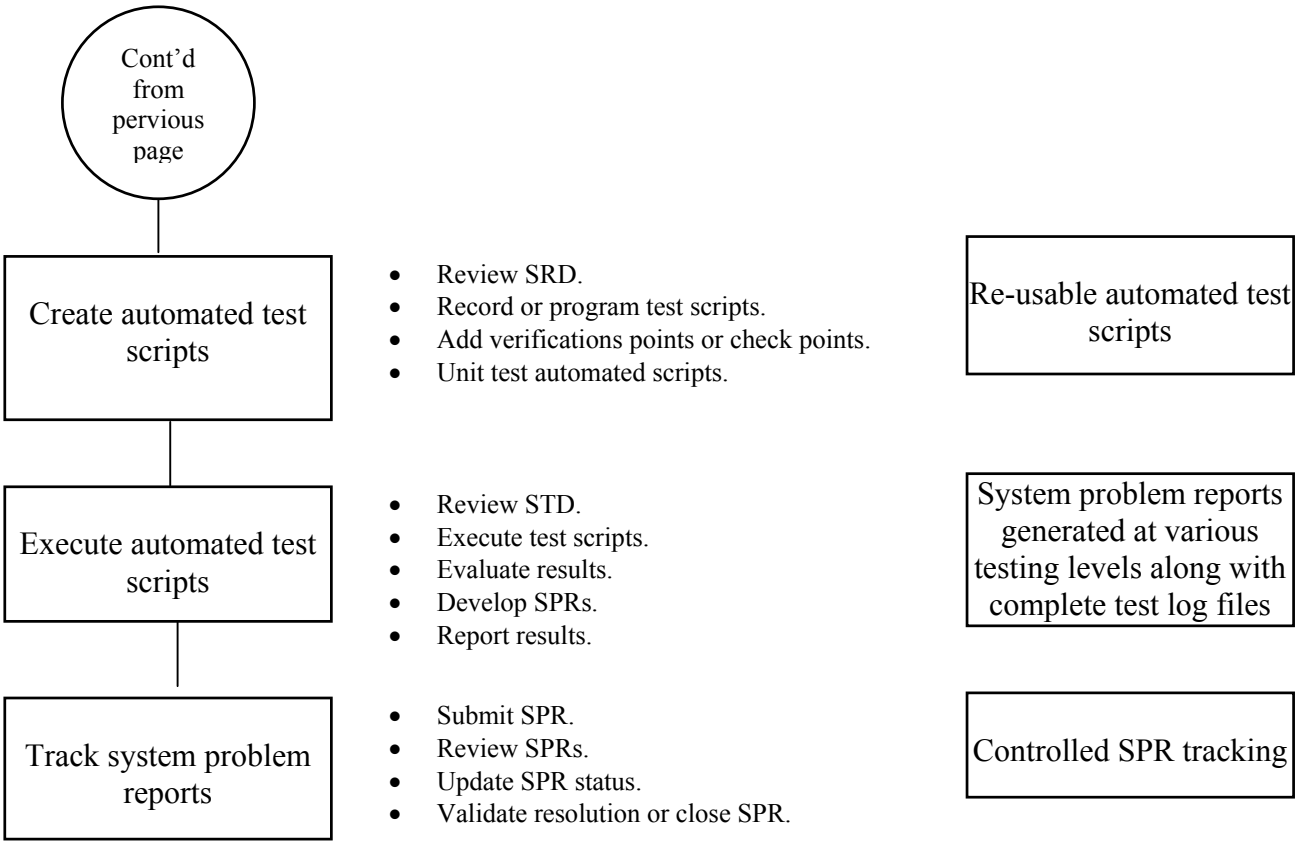


Figure 2-1. Procedures For Using An Automated Test Tool

2.2 Configuring the Test Tool

Prior to using a test tool, it is necessary to plan the size, organization, and access rights to the test resources. Once test tool configuration planning has been completed, the selected test tool must to be customized to meet the reporting requirements of project management.

On large projects, it may be necessary to further sub-divide project definitions to more accurately represent system development areas of responsibility. One large project may have several repositories and project definitions. Each of these repositories and project definitions will have specific users with the ability to add, change, or delete information within the test tool database(s).

A test tool is used to control and organize test resources. The storage space used to control the resources is defined as a repository. A project identification field defined within the repository is used to organize test resources.

2.2.1 Participants

a. Project Manager.

b. Test Manager.

c. Systems administrator for repository server(s).

d. Testing tool software administrator(s).

2.2.2 Entrance Criteria

a. The scope of the testing, composition of the test teams, and defined reporting requirements.

b. The new Hardware Configuration Item (HWCI) needed for automated testing has been procured, or the necessary HWCI capacity is available and reserved.

2.2.3 Inputs

Approved test plan documented in the baseline STD and STP documents.

2.2.4 Activities

a. Obtain required software licenses.

b. Configure HWCI, as required.

c. Establish access and roles for test tool users.

d. Create shared directories on server(s).

e. Create repository within the test tool.

f. Create and maintain test tool database(s).

g. Identify management-defined reporting fields within test tool database(s).

h. Create standard reports to be shared with project management.

2.2.5 Outputs

a. Controlled test tool configuration.

b. Standard report formats.

2.2.6 Exit Criteria

a. All test team personnel can enter information into the test tool database(s).

b. Appropriate users and administrators can modify information, if necessary.

c. Project management approval of report formats.

2.3 Defining Software Requirements

Requirements are a capability or condition to which the system under test must conform. Software requirements are defined in the SRD. Specific requirements to be tested through the SI&T process are extracted and form the basis of the STD.

After the requirements are determined for the purpose of creating the STD, they can be defined and organized within a test tool. Controlling requirements through a test tool is a methodology of organization and documentation. The control of requirements with a test tool provides a means to accomplish the following:

a. Maintaining agreement between the customer and the development team regarding any changes made to the software requirements.

b. Maintaining agreement between the System Development (SD) group and the test group regarding scope and expected results of tests.

c. Tracing software requirements defined in the SRD to test cases defined in the STD.

A sample software requirement tracking table layout is presented in Appendix C.

2.3.1 Participants

a. Participants will include government personnel designated/assigned by the SFA and/or the SI&T Project Manager. Participants will oversee activities performed and provide documentation and answers to the SI&T team (contractor and/or government). Participants should have knowledge and/or understanding of the system to be tested.

Participants may include the Project Manager of the system being tested, a test manager and/or engineer, a QA group or person, SD group or person, and a CM group or person. The size and composition of the participating group will vary from system to system due to system complexity, size, function, etc.

b. Technical writer(s).

2.3.2 Entrance Criteria

a. Approved software requirements contained in the baseline SRD.

b. Software requirements documented in the baseline STP.

2.3.3 Inputs

a. Baseline STP.

b. Baseline SRD.

2.3.4 Activities

a. Review software requirements documented in the SRD.

b. Enter software requirements documented in the SRD into the test tool database. At a minimum the following information must be included in each entry:

1) A unique identifier.

2) The software requirement.

3) The author of the requirement.

4) Fields for producing reports defined by project management.

c. Review the software requirements entered into the test tool database.

d. Add to or modify, as necessary, the entered software requirements.

e. Approval of the software requirements in the test tool database.

f. Place the approved software requirements under CM.

g. Produce proposed testing reports for review.

h. Review and approve proposed testing reports.

2.3.5 Outputs

Approved software requirements defined within the test tool.

2.3.6 Exit Criteria

a. All necessary SRD-defined software requirements entered into the test tool.

b. Approved software requirements placed under CM.

c. Approval of proposed testing reports.

2.4 Defining Test Cases And Test Requirements

The STD describes test preparations, test cases, and test case procedures used to perform testing during the Integration Test, Performance Test, and SQT phases of the SI&T process. The STD enables everyone involved with a project to assess the adequacy of the testing to be performed.

After the test cases are approved for the STD, they are then defined to the test tool as test requirements. Using the test tool to control the test requirements is a methodology that allows optimum organization, control, and documentation of the test requirement information. A methodology adds to the ease with which CM and QA can track test requirements and provides test personnel with the ability to trace test requirements to software requirements defined in the SRD.

A sample test requirement tracking table layout is presented in Appendix C.

2.4.1 Participants

a. Participants will include government personnel designated/assigned by the SFA and/or the SI&T Project Manager. Participants will oversee activities performed and provide documentation and answers to the SI&T team (contractor and/or government). Participants should have knowledge and/or understanding of the system to be tested.

Participants may include the Project Manager of the system being tested, a test manager and/or engineer, a QA group or person, SD group or person, and a CM group or person. The size and composition of the participating group will vary from system to system due to system complexity, size, function, etc.

b. Technical writer(s).

2.4.2 Entrance Criteria

Baseline STD identifying test cases based on system functionality.

2.4.3 Inputs

Baseline STD.

2.4.4 Activities

a. Review test requirements documented in the STD.

b. Enter test requirements documented in the STD into the test tool database. At a minimum the following information must be defined during entry:

1) A unique identifier.

2) The test requirement.

3) The author of the test requirement.

4) Required fields for producing reports, as defined by project management.

c. Review test requirements entered into the test tool database(s).

d. Add to or modify, as necessary, the entered test requirements.

e. Approval of the test requirements entered into the test tool database.

f. Place the approved test requirements under CM.

g. Produce proposed test reports for review.

h. Review and approve proposed test reports.

2.4.5 Outputs

Approved test requirements defined within the test tool.

2.4.6 Exit Criteria

a. All STD test requirements defined within the test tool.

b. Approved test requirements are placed under CM.

2.5 Procedures For Creating Automated Test Scripts

Section 4.2.1.6 of the baseline STD defines detailed steps for each test case. Test cases that should be automated are determined from the STD. Automated test scripting is the recording or programming of test case procedures defined by a test case. A test engineer creates an automated test script that implements the test case procedures. The automated test script references a unique identifier of the test case in the description, header, or name of the automated test script.

Prior to the start of formal integration testing, the automated test script must be tested and validated. Testing the automated test script will provide a clear baseline test log file for verification of future test results.

2.5.1 Participants

a. Participants will include government personnel designated/assigned by the SFA and/or the SI&T Project Manager. Participants will oversee activities performed and provide documentation and answers to the SI&T team (contractor and/or government). Participants should have knowledge and/or understanding of the system to be tested.

   Participants may include the Project Manager of the system being tested, a test manager and/or engineer, a QA group or person, SD group or person, and a CM group or person. The size and composition of the participants will vary from system to system due to system complexity, size, function, etc.

b. Technical writer(s).

2.5.2 Entrance Criteria

Approved test cases identified in baseline STD.

2.5.3 Inputs

a. Baseline STD.

b. Test requirements allocated to test cases.

c. Available user and/or operator manuals or checklists.

2.5.4 Activities

a. Review software user and operator manuals to identify methods of operator(s) input(s), use of simulator or emulator tools, and software data recording.

b. Define automated test scripts to provide inputs for test requirements.

c. Define measurable detailed test results for automated test script(s). These defined test results become the automated checkpoints or verification points that indicate Pass/Fail status of associated test requirement.

d. Prepare input data files to provide proper test stimuli.

e. Conduct a peer review.

f. Revise verification points or check points to correct discrepancies, then incorporate the recommended changes.

g. Record or program automated test scripts with test tool.

h. Execute automated test scripts that provide test log files.

2.5.5 Outputs

a. Documented automated test scripts that are traceable to STD test cases.

b. Input and expected test results for each automated test script.

c. Compiled automated test scripts.

d. Baseline test log files.

2.5.6 Exit Criteria

a. Baseline automated test scripts.

b. Baseline test log files.

2.6 Procedures For Executing Automated Test Scripts

Test engineers will execute automated test scripts based upon a test execution schedule. Regardless of the testing phase, automated test scripts, must be executed using exactly the same procedures. Using the same procedures ensures repeatability and allows for comparison of baseline test logs. Testing phases are further described in the Department of Education, SFA, document "Procedures and Templates for Test Execution" dated November 2000.

The test engineers will conduct evaluation activities regarding the results of automated test script execution to avoid false-positive or false-negative test results. System problems encountered are documented with a SPR. Sufficient documentation is provided with the SPR to support understanding of system problems and replication of the problem(s) by the software developer. The documentation supporting a SPR should contain screen prints or copies of output files, etc.

2.6.1 Participants

a. Test engineer.

b. Witnesses (as required).

2.6.2 Entrance Criteria

a. A defined test environment, under CM, established in baseline STP.

b. Verification of approved HWCI and software or CSCI configuration.

c. Automated test scripts developed in accordance with established test requirements in STD.

2.6.3 Inputs

a. Software to be tested under CM.

b. Baseline test requirements written for individual test cases.

c. Approved automated test scripts.

d. Expected test results and Pass/Fail criteria established in the STD.

2.6.4 Activities

a. Verify that test results are available.

b. Conduct pre-test inspections of hardware and software configurations and the test environment.

c. Execute test(s) according to the STD.

d. Preserve test log files, as described in the STP.

e. Analyze Pass/Fail criteria results.

f. Submit SPR, if necessary.

g. Provide documentation to support test failures (e.g., screen prints, printed reports, etc.).

2.6.5 Outputs

a. SPRs, with supporting documentation, for tests that did not achieve Pass criteria.

b. Compiled automated test scripts.

c. Baseline test log files.

2.6.6 Exit Criteria

a. Approved baseline procedures for all test cases in the baseline STD.

b. Baseline test log files.

2.7 Procedures For Tracking System Problem Reports

Report system problems as they are discovered. Once a problem is discovered, an SPR is created. The SPR is then entered into a tracking database and controlled throughout the life of the project. The SPR tracking process is depicted in Figure 2.2.

Submitted SPRs pass through a formal review process. Representatives from each group in the system test organization participate in the SPR review. Representatives from the SD group and the test group jointly determine the risk each problem poses. An assessment is made to determine the severity of the problem and the priority is assigned for repairing the problem.

The SPR review group uses priorities to express the level of urgency for repair of the problem. The definition of critical and non-critical system defects or problems should be addressed at a management level and can be different for each system. For any given system error, defect, problem, or discrepancy, an appropriate impact value (i.e., priority) will be assigned.

An example of impact values with the corresponding priority numbers is presented below as contained in IEEE/EIA Std-12207, 1998.

The priority that will apply if a problem can result in one or more of these impacts:

PRIORITY	IMPACT
1.	a.) Prevent the accomplishment of an operational or mission essential capability.
	b.) Jeopardize safety.
	c.) Cause significant technical, cost, or schedule risks to the project or to life cycle support of the system.
2.	a.) Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is known.
	b.) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around is known.
3.	a.) Adversely affect the accomplishment of an operational or mission essential capability, but a work-around solution is known.

b.) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around is known.

4. a.) Results in user/operator inconvenience or annoyance, but does not affect a required operational or mission essential capability.

b.) Results in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of the responsibilities of these personnel.

5. a.) This priority denotes any other effect.

Priority is a measure of the impact and elements of importance that occur during a test process. The impact and elements of importance are considered in defining the severity of a system problem. In the SI&T process, severity is defined as the degree to which a problem adversely influences system operation or the overall test effort. The effect of a problem can range from extreme (e.g., data loss or hardware damage) to minimal (e.g., cosmetic display screen error). Both the SD group and the Test group must review and agree on the level of severity. Severity is represented in terms of high, medium, or low.

During the SPR review, each SPR is assigned to the SD group, identified as a duplicate of a previously reported SPR, or postponed until a later date for SPR review. The SPR review process is not limited to new submissions or those SPRs previously postponed. The SPR review must also consider whether SPRs are being processed appropriately and expeditiously.

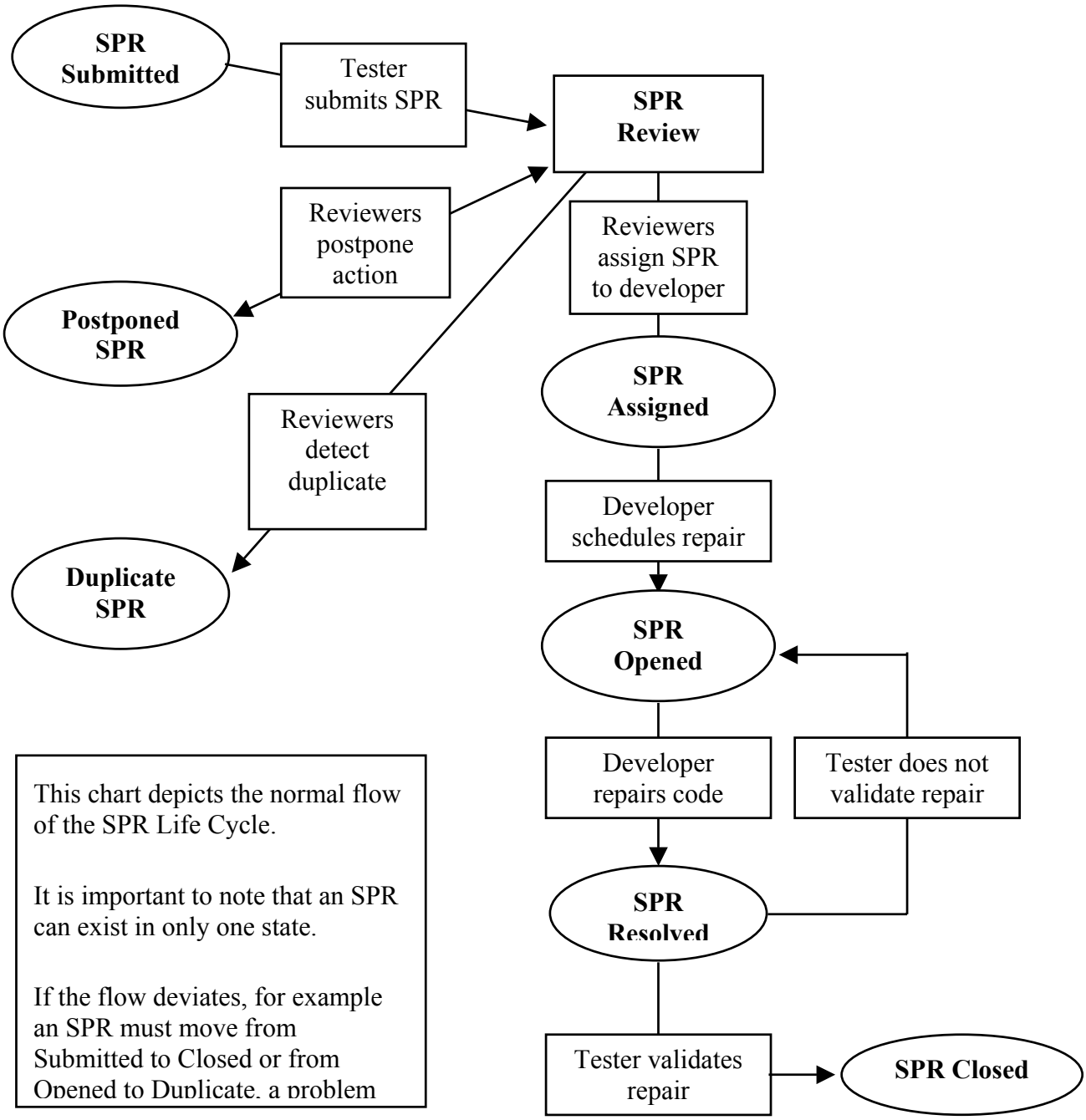


Figure 2.2 System Problem Report Tracking

During the SPR review, each SPR is assigned to the SD group, identified as a duplicate of a previously reported SPR, or postponed until a later date for SPR review. The SPR review process is not limited to new submissions or those SPRs previously postponed. The SPR review must also consider whether SPRs are being processed appropriately and expeditiously.

During its life cycle, an SPR will reside in one of the following states:

- Submitted
- Postponed
- Duplicate
- Assigned
- Opened
- Resolved
- Closed

The time a SPR remains in a specific state is used to assess the quality of the SPR identification and resolution process. It is for this reason that duplicates are identified and remain in that state. Duplicate SPRs are only used for informational purposes and are not "Closed."

In order to facilitate the tracking process and support testing report requirements, the following data elements must be captured from the SPR form upon submission:

- A unique identifier for the SPR.
- A problem description.
- The specific test case and software version in which the problem was discovered.
- The date on which the problem was reported.
- The test engineer or observer who reported the problem.

As the SPR is tracked through its life cycle, more information is added to the tracking database. The following data elements are added or changed with progression:

- The current state of the SPR
- The date SPR was placed in its current state.
- The date on which the SPR was last reviewed.
- The member of the development team to whom the SPR was assigned.
- The date on which the developer opened the SPR for analysis.
- The date on which the developer changed the state to resolved.
- The date on which a test engineer last tested for validation of resolution.
- The date on which the test engineer returned the item to the "Opened" state because the item failed re-test and resolution could not be validated.
- The severity of the problem, as determined by the SPR review process.
- The priority for repair of the problem, as determined by the SPR review process.

A recommended database layout is presented in Appendix B.

2.7.1 Participants

Participants will include government personnel designated/assigned by the SFA and/or the SI&T Project Manager. Participants will oversee activities performed and provide documentation and answers to the SI&T team (contractor and/or government). Participants should have knowledge and/or understanding of the system to be tested.

Participants may include the Project Manager of the system being tested, a test manager and/or engineer, a QA group or person, SD group or person, and a CM group or person. The size and composition of the participating group will vary from system to system due to system complexity, size, function, etc.

2.7.2 Entrance Criteria

a. Tests were executed, and the results were analyzed.

b. System problems were identified.

c. A SPR form was submitted.

2.7.3 Inputs

a. Software under CM.

b. A SPR that is recommended for change of status.

c. A newly created SPR form.

2.7.4 Activities

a. Enter the SPR into the tracking database.

b. Review outstanding SPRs, and if necessary, gather additional documentation and evidence to support the validity of each SPR.

c. Participate in formal SPR review session(s) to determine SPR disposition.

d. Update the SPR database for new SPRs with a status of Assigned, Duplicate, or Postponed.

e. Update the SPR database for existing SPRs with a status of Open, Resolved, or Closed.

f. Develop report(s) detailing the effectiveness of problem identification and resolution.

2.7.5 Outputs

SPR status reports.

2.7.6 Exit Criteria

Only SPRs within the tolerance of the SRD remain in the tracking system. These SPRs are identified in the System Test Report (STR).

GLOSSARY

Aggregate

A mass of distinct things gathered into a total or whole.

Aggregation Level

Effective measurement analysis and reporting requires that the data be aggregated to higher levels of the of the software components and project organizational structure. The aggregation levels define the different ways the measurement data can be grouped and organized for reporting on the project. The aggregation levels describe how the measurement data relates to an existing product and process structures. The organization that allows the measurement results to be combined, and later decomposed, into meaningful pieces of information.

Aggregation Structure

The structure used to define the data according to the defined aggregation levels. The levels may describe the personnel and management structure of the project, or the configuration of physical components of the project. All entries in a structure should be of the same type, such as software modules. However, these entries may reside at various levels of the structure, such as software modules at the unit level, CSCI, or integrated level of the software architecture.

Application

(1.) A complete, self-contained program that performs specific function(s) directly for the user.

(2.) In the TPM process this term refers to one of the two basic measurement activities which comprise the system measurement process. The application activity involves collecting, analyzing, and acting upon the measurement data.
See **Tailoring**.

Automated Test Script

A computer readable set of instructions that performs a sequence of steps, sub-steps, or other actions, performed serially, in parallel, or in some combination of consecution, that creates the desired test conditions that the test case is deigned to evaluate.

Baseline

A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

Baseline Control

Baseline control is the process that regulates approved and released versions of all software, documentation, and the test environment throughout the test life cycle.

Black Box Testing

This is testing associated with functional testing where the object being tested is treated as a black box. In this type of testing the test object is subjected to inputs and outputs that are verified for conformance to prescribed specifications.

Capacity Testing

Attempts to simulate expected customer peak load operations in order to ensure that the system performance requirements are met. It does not necessarily exercise all of the functional areas of the system, but selects a subset that is easy to replicate in volume. It will ensure that functions which are expected to use the most system resources are adequately represented.

Change Control

The process by which problems and changes to the software, documentation, and test environment are evaluated, approved, rejected, scheduled, and tracked.

Computer Aided Software Engineering (CASE)

A technique for using computers to help with one or more phases of the software life cycle, including the systematic analysis, design, implementation and maintenance of software. Adopting the CASE approach to building and maintaining systems involves software tools and training for the developers who will use them.

Computer Software Configuration Item (CSCI)

An aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

Configuration Control

An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.

Configuration Item (CI)

Hardware or software, or an aggregate of both, which is designated by the project configuration manager (or contracting agency) for configuration management.

Configuration Management (CM)

A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

Configuration Management Office (CMO)

The Configuration Management Office (CMO) is the officiator of the project life cycle CM process.

Criteria

A standard, rules, or tests by which something can be judged.

Critical Defect

See Criticality

Criticality

The assessment of the impact upon a system of a given error, defect, problem, or discrepancy during the life cycle of a system.

The definition of critical and non-critical system defects or problems should be addressed at a management level and can be different for each system. For any given system error, defect, problem, or discrepancy, an appropriate impact value (i.e., priority) will be assigned.

An example of impact values with the corresponding priority numbers is presented below as contained in IEEE/EIA Std-12207, 1998. The priority that will apply if a problem can result in one or more of these impacts:

PRIORITY	IMPACT
1.	a.) Prevent the accomplishment of an operational or mission essential capability.
	b.) Jeopardize safety.
	c.) Cause significant technical, cost, or schedule risks to the project or to life cycle support of the system.
2.	a.) Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is known.
	b.) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around is known.
3.	a.) Adversely affect the accomplishment of an operational or mission essential capability, but a work-around solution is known.
	b.) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around is known.
4.	a.) Results in user/operator inconvenience or annoyance, but does not affect a required operational or mission essential capability.
	b.) Results in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of the responsibilities of these personnel.
5.	a.) This priority denotes any other effect.

Customer

The organization that procures software systems for itself or another organization.

Developer

An organization that develops software products. The term “develop” may include develop, modification, integration, reengineering, sustaining engineering, maintenance, or any other

activity that results in software products. The developer may be a contractor or a government agency.

Discrepancy

An inconsistency or disagreement found during testing between the actual and expected test results.

Document

A data medium and the data recorded on it that generally has permanence and can be read by a human operator or machine. Often used to describe human readable items only (e.g., technical documents, design documents, requirements documents, etc.).

Documentation

(1.) A collection of documents on a given subject.

(2.) The management of documents, that includes the actions of identifying, acquiring, processing, storing, and disseminating.

(3.) Any written or pictorial information describing, defining, specifying, reporting or certifying activities, requirements, procedures, or results.

Driver

A software program that exercises a system or system component by simulating the activity of a higher level component.

Emulation

One system is said to emulate another when it performs in exactly the same way, though perhaps not at the same speed. A typical example would be the emulation of one computer by (a program running on) another. You might use emulation, as a replacement for a system whereas you would use a simulation if you just wanted to analyze it and make predications about it.

Emulator

Hardware or software that performs emulation.

Entry Criteria

A set of decision making guidelines used to determine whether a system under test is ready to move into, or enter, a particular phase of testing. Entry criteria tend to become more rigorous as the test phases progress.

Environment

The infrastructure in which a system is executing, consisting of hardware, operating system software, interfaces, etc.

Exit criteria

A set of decision-making guidelines used to determine whether a system under test is ready to exit a particular phase of testing. When exit criteria are met, either the system under test moves on to the next test phase or the test project is considered complete. Exit criteria tend to become more rigorous as the test phases progress.

Final System Test Report (FSTR)

Used to determine whether system testing is completed and to assure that software is ready for production.

Hardware Configuration Item (HWCI)

An aggregation of hardware that is designated for configuration management and treated as a single entity in the configuration management process.

Independent Verification and Validation (IVV)

The verification and validation of a software product by an organization that is both technically and managerially separate from the organization responsible for developing the product.

Indicator

A measure or combination of measures that provides insight into a system issue or concept. TPM frequently uses indicators that are comparisons, such as planned versus actual measures. Indicators are generally presented as graphs or tables.

Integration

Combining software or hardware components or both into an overall system.

Integration Testing

The period of time in the software lifecycle during which the application is tested in a simulated production environment to validate the communications and technical architecture of the system. This test phase occurs when all the constituent components of the system under test are being integrated.

Interactive Development Environment (IDE)

A system for supporting the process of writing software. Such a system may include a syntax-directed editor, graphical tools for program entry, and integrated support for compiling and running the program and relating compilation errors back to the source code.

Interface

(1.) A shared boundary (e.g., a hardware component linking two devices or registers, or a portion of storage accessed and/or modified by two or more computer programs).

(2.) To interact or communicate with another system component.

Interface Requirement

A requirement that specifies a hardware, software, or database element with which a system or system component must interface, or that sets forth constraints caused by such an interface.

Interface Specification

A specification that sets forth the interface requirements for a system or system component (e.g., the software interface specification document).

Interface Testing

Tests conducted to ensure that program or system components correctly pass data and/or control to one another.

Issue

An area of concern where obstacles to achieving program objectives might arise. Issues include risks, problems, and lack of information. These three types of issues are defined as:

- Risk -- An area of concern that could occur, but is not certain. A risk is a potential problem. Risks represent the potential for the realization of unwanted, negative consequences from a project event. For example, a project plan may be based on the assumption that a COTS component will be available on a given date. There is a possibility (probability) that the COTS may be delayed and have some amount of negative impact on the project.

- Problem -- An area of concern that a project is currently experiencing or is relatively certain to experience. For example, a shortage of staff with the right skills may be an actual problem that is delaying the project.

- Lack of Information -- An area where the available information is inadequate to reliably predict project impact. Thus, satisfaction of project objectives is questionable even if no problems or risks are present. For example, lack of information about the size of the software to be developed could result in the project "discovering" that it has more work to do than originally planned.

Measure

The result of counting or otherwise quantifying characteristics of a process or product. Measures are numerical values assigned to system attributes according to defined criteria.

Measured (or actual) Value

Actual, current measurement data, such as hours of effort expended or line of code produced.

Measurement

The process of assigning quantitative values of system properties, according to some defined criteria. This process can be based on estimation or direct measurement. Estimation defines planned or expected measures. Direct measurement results in actual measures.

Measurement Analysis

The uses of measurement data to identify problems, assess problem impact, project an outcome, or evaluate alternatives related to system issues.

Measurement Analyst

The person(s) or team responsible for tailoring and applying system measures for a given project or task.

Measurement Information

Knowledge derived from analysis of measurement data and measurement indicators.

Milestone

A scheduled event for which some project or task member or manager is held accountable. A milestone is often used to measure progress.

Module

A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading.

Note: *The terms 'module', 'component', and 'unit' are often used interchangeably or defined to be sub-elements of one another in different ways depending on the context.*

Non-Critical Defect

See Criticality

Performance Testing

The period of time in the system or software development lifecycle during which the response times for the application are validated to be acceptable. The tests ensure that the system environment will support production volumes, both batch and on-line.

Priority

A measure of the elements of importance related to the repair of a system problem that are not considered in defining the severity of a system problem.

Project Manager (PM)

The official responsible for acquiring, developing, or supporting a system to meet technical, cost, schedule, and quality requirements. Acquisition, development, and support will include both internal tasks and work that is contracted to another source.

Quality Assurance (QA)

A planned and systematic pattern of all actions necessary to provide adequate confidence that the product optimally fulfils customers expectations.

Quality Control (QC)

The assessment of product compliance. Independently finding deficiencies assures compliance of the product with stated requirements.

Requirement

(1.) A condition or capability needed to solve a problem or achieve an objective.

(2.) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. The set of all requirements forms the basis of development.

Regression testing

Part of the test phase of software development where, as new modules are integrated into the system and the added functionality is tested, previously tested functionality is re-tested to assure that no new module has corrupted the system.

Risk

An area of concern that may occur, but is not certain. A risk is a potential problem. Risks represent the potential for the realization of unwanted, negative consequences from a project event. For example, a project plan may be based on the assumption that a commercial off the shelf (COTS) component will be available on a given date. There is a possibility (probability) that the COTS may be delayed and have some amount of negative impact on the project.

Severity

The degree to which a problem adversely influences the system's operation or the overall test effort.

Simulation

Attempting to predict aspects of the behavior of a system by creating an approximate (mathematical) model of it. This can be done by physical modeling, by writing a special-purpose

computer program or using a more general simulation package, aimed at a particular kind of simulation. Typical examples are aircraft simulators or electronic circuit simulators.

Simulator

Hardware or software that performs simulation.

Software Design Specification (SDS)

A document that records the design of a system or system component; typical contents include: system and/or component algorithms, control logic, data structures, data set use, input/output formats, and interface descriptions.

Software Development File (SDF)

The developer shall document the development of each Computer Software Unit (CSU), Computer Software Component (CSC), and CSCI in Software Development Files (SDF). The developer shall establish a separate SDF for each CSU or a logically related group of CSUs, for each CSC or a logically related group of CSCs, and for each CSCI. The developer shall document and implement procedures to establish and maintain SDFs. SDFs may be generated, maintained, and controlled by automated means. To reduce duplication, SDFs should not contain information provided in other documents or SDFs. The set of SDFs shall include (directly or by reference) the following information:

- Design considerations and constraints.
- Design documentation and data.
- Scheduling and status information.
- Test requirements and responsibilities.
- Test case, test case procedures, and results.

Software Life Cycle

The phases a software product goes through between when it is conceived and when it is no longer available for use. The software life cycle typically includes the following: requirements, analysis, design, construction, testing (validation), installation, operation, maintenance, and retirement. The development process tends to run iteratively through these phases rather than linearly; several models (spirals, waterfall, etc.) have been proposed to describe this process.

Other processes associated with a software product are: quality assurance, marketing, sales, and support.

Software Management Plan

A project plan for the development of the software component of a system or for the development of a software product.

Software Requirements Document (SRD)

This is a formal document derived from the Software Requirements Specification (SRS) that sets forth the requirements, specifications, and standards for a system (e.g., a software product). Typically included are functional specifications and requirements, performance specifications and requirements, interface specifications and requirements, design specifications and requirements, and development requirements and standards.

Software Requirements Specification (SRS)

A specification that sets forth the requirements for a system component; (e.g., a software product). Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards.

Software Tool

Computer programs used to help develop, test, analyze, or maintain another computer program or its documentation.

Specification

Documentation containing a precise, detailed, verifiable description of particulars with respect to the requirements, design, function, behavior, construction, or other characteristics of a system or system component.

Stub

(1.) A dummy procedure used when linking a program with a run-time library. The stub routine need not contain any code and is only present to prevent "undefined label" errors at link time.

(2.) A local procedure in a remote procedure call (RPC). The client calls the stub to perform some task and need not necessarily be aware that RPC is involved. The stub transmits parameters over the network to the server and returns the results to the client/caller.

System

(1.) Any large program.

(2.) The entire computer system, including the input/output devices, supervisor program or operating system and possibly other software.

System Problem Report (SPR)

A form that is used to record a discrepancy discovered during the Integration Test, Performance Test and System Qualification Test phases of the SI&T process concerning a Computer Software Configuration Item, a software system or subsystem, other software related items, and associated documentation.

System Problem Report (SPR) Status Report

The System Problem Report Status Report is used during the SPR Status Review to determine if the SPRs are being processed appropriately and expeditiously.

System Testing

The period of time in the software lifecycle during which the implementation of each requirement is validated.

Tailoring

In the TPM process, this term refers to one of the two basic measurement activities, which comprise the system measurement process. The tailoring activity includes identification and prioritization of program issues, selection and specification of appropriate system measures, and integration of the measurement requirements to the developer's system process.
See **Application**.

Test

The process of exercising a product to identify differences between expected and actual behavior.

Test Artifacts

An item created during the system integration and test process that is preserved upon completion of the test process (e.g., test plans, requirements documentation, automated test scripts, and test documentation).

Test Case

A description of a test to be executed for or focused on a specific test aim.

Test Case Procedures

A sequence of steps, sub-steps, and other actions, performed serially, in parallel, or in some combination of consecution, that creates the desired test conditions that the test case is designed to evaluate.

Test Case (Setup) Suite

The steps required to configure the test environment for execution of a test case.

Testing Condition

System state or circumstance created by proceeding through some combination of steps, sub-steps, or actions in a test case.

Testing Environment

The infrastructure in which the test is performed, consisting of hardware, system software, test tools, and procedures.

Test Plan

In a test plan the general structure and the strategic choices with respect to the test to be executed are formulated. The test plan forms the scope of reference during execution of the test and also serves as an instrument to communicate with the customer of the test. The test plan is a description of the test project, including a description of the activities and planning, therefore it is *not* a description of the tests themselves.

Test Readiness Review (TRR)

Review conducted to determine whether a software test phase has been completed and to assure that the software is prepared for the next step in the formal integration and testing procedures. Software test procedures and results are evaluated, for compliance with the software testing requirements and system descriptions, for adequacy in accomplishing testing goals. Also, provides the forum for updating and revising operational and supporting documentation.

Test Resources

Aids that are used by a test tool for collecting, tracking and controlling information. This information is:

- Software requirements defined in the Software Requirements Document.
- Test requirements defined in the System Test Description.
- Automated test case scripts as defined in the System Test Description.
- SPRs as determined at each phase of the System Integration and Testing process.

This information is controlled by Configuration Management at the end of the SI&T process for use whenever further testing may be conducted, using a testing tool, during the remaining lifecycle of the software or system.

Test Tools

The software, hardware, systems, or other instruments that are used to measure and test an item.

Traceability

Degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor, successor, or master-subordinate relationship to one another (e.g., the degree to which the requirements and design of a given software component match).

Unit

The lowest element of a software hierarchy that contains one or more of the following characteristics:

(1.) A unit comprising one or more logical functional entities.

(2.) An element specified in the design of a computer software component that is separately testable.

(3.) The lowest level to which software requirements can be traced.

(4.) The design and coding of any unit can be accomplished by a single individual within the assigned schedule.

Unit Test

The process of ensuring that the unit executes as intended. This usually involves testing all statements and branch possibilities.

Version

One of a sequence of copies of a system, each incorporates new modifications.

Version Identifier

A unique identifier assigned to baseline software, documentation, and test environment.

Version Control

The process by which all changes to the software, documentation, and test environment are compiled and built into a new version of the system.

Version Control Report

A report that details all changes and enhancements made to current version of the software, documentation, and test environment.

White Box Testing

This type of testing is associated with structural testing in which the testing can be characterized as being tied to implementation details, such as control methods, database design, coding details, and logic paths. The process of how an individual input is treated to produce a given output is ascertained. Structural testing is sometimes referred to as "clear box testing" since white boxes are considered opaque and do not really permit visibility into the code.

Work Breakdown Structure (WBS)

A work breakdown structure for software defines the software-related elements associated with program work, work activities, and products. Many measures are aggregated and analyzed at various WBS levels.

BIBLIOGRAPHY

Black, Rex. *Managing The Testing Process*, Redman, WA: Microsoft Press, 1999

Koomen, Tim and Pol, Martin. *Test Process Improvement*, Essex, England, UK: Pearson Education Limited, 1999

Evans, Michael and Marciniak, John. *Software Quality Assurance and Management,* New York, NY: John Wiley & Sons, 1987

Carnegie Mellon University, Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*, 1995.

Institute of Electrical and Electronics Engineers (IEEE). "Glossary of Software Engineering Terminology," IEEE-Std-610.12, 1990.

Institute of Electrical and Electronics Engineers (IEEE)/Electronic Industries Alliance (EIA). "Software Life Cycle Processes," IEEE/EIA Std-12207, 1998.

U.S. Department of Education. "SFA System Integration & Testing Approach, SFA Modernization," Undated.

U.S. Department of Education, SFA. "Procedures and Templates for Test Creation", November 2000.

Federal Systems Integration and Management Center (FEDSIM). "FEDSIM Writers Guide, Version 2," May 1994

Free On-Line Dictionary Of Computing web site at www.foldoc.org

This page intentionally left blank

APPENDIX A

TESTING TOOL PROCEDURES CHECK-OFF LIST

This page intentionally left blank

TESTING TOOL PROCEDURES CHECK-OFF LIST TEMPLATE

Version 1.0

November 24, 2000

This page intentionally left blank

FOREWORD

This template was developed to provide a check-off list for use with phases of system testing that involve use of an automated testing tool. This template should be supplemented with system-specific information to produce a check-off list that accurately enables management, support, and technical personnel to assess and analyze planned procedures.

The U. S. Department of Education, SFA will retain and maintain this Testing Tool Procedures Check-Off List template. Document users may report deficiencies in and, or corrections to, this document using the Document Change Request (DCR) form. U.S. Department of Education, SFA, collects and processes reported information as inputs for process improvements to the Testing Tool Procedures Check-Off List template.

This page intentionally left blank

DOCUMENT CHANGE REQUEST (DCR)

DOCUMENT: Testing Tool Procedures Check-Off List Template, Version 1.0

SFA TRACKING NUMBER:_____

NAME OF SUBMITTING
ORGANIZATION:_____

_____

ORGANIZATION CONTACT:_____TELEPHONE:_____

MAILING ADDRESS:_____

_____

DATE:_____ SHORT TITLE:_____

CHANGE LOCATION:_____
(Use section #, figure #, table #, etc.)

PROPOSED CHANGE:

REASON FOR CHANGE:

.....

Note: For the U. S. Department of Education, Office of Student Financial Assistance (SFA), to take appropriate action on a change request, please provide a clear description of the recommended change along with supporting reason.

Send to: U. S. Department of Education, Office of Student Financial Assistance, 400 Maryland Avenue SW, Washington DC 20202 or Fax to: (202) 205-8532.

DCR Form November 2000

This page intentionally left blank

RECORD OF CHANGES

CHANGE NUMBER	DATE	FIGURE, TABLE, OR PARAGRAPH NUMBER	**A/M/D***	TITLE OR BRIEF DESCRIPTION	CHANGE REQUEST NUMBER

* **A** = ADD
M = MODIFY
D = DELETE

This page intentionally left blank

DOCUMENT CONVENTIONS

NOTE: *The next page is the start of the template. Delete this page and preceding pages before final submission format of your project or system Testing Tool Procedures Check-Off List*

This page intentionally left blank

U. S. DEPARTMENT OF EDUCATION
OFFICE OF STUDENT FINANCIAL ASSISTANCE

TESTING TOOL PROCEDURES CHECK-0FF LIST FOR [PROJECT NAME]

U. S. Department of Education
Office of Student Financial Assistance
400 Maryland Avenue, SW
Washington DC 20202

This page intentionally left blank

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 1	**Planning and Configuring the Test Tool**		
Step 1	Establish a Repository		
Task 1	Hardware Resources Appropriate resources must be secured to accommodate the following: 1. Test tool software		
	2. Test tool software configuration files		
	3. Software requirements database		
	4. Test requirements database		
	5. Test scripts		
	6. SPRs tracked		
	7. Documentation for completed tests		
Task 2	Test Tool Access The following information must be planned: 1. Who will be the system administrator(s)		
	2. Who will administer test tool database(s)		
	3. Who will be assigned as users of test tool		
	4. Are software licenses in place for users		
	5. How will users access test tool		
Task 3	Directory Creation 1. System administrator establishes shared directories		
Task 4	Repository Creation 1. Test tool administrator defines repositories within the test tool		

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 1	**Planning and Configuring the Test Tool**		
Step 2	Establishing Project Reporting Criteria		
Task 1	Define Project Reporting Fields Appropriate fields must exist to accommodate the following: 1. Tracing test resource creation to the project schedule		
	2. Tracing test resource to system development area of responsibility		
	3. Tracing test requirements to software requirements		
	4. Tracing test scripts to test requirements (implicitly to software requirements)		
	5. Tracking all information on an System Problem Report (SPR)		
	6. Tracing SPR to test requirements (implicitly to software requirements)		
	7. Tracing SPR to system development area of responsibility		
Task 2	Configure Users 1. Test tool administrator defines the users (and their permissions) to the test tool		
Task 3	Create Standard Report Formats 1. Test tool administrator designs a set of standard reports for reporting the progress of testing and the disposition of SPRs		
Task 4	Review Standard Report Formats 1. Project management reviews a set of standard reports for reporting progress of testing and disposition of SPRs		

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 2	**Software Requirements Tracking**		
Step 1	Controlling Software Requirements Resources		
Task 1	Enter Software Requirements Software Requirements are a product of the Software Requirements Document (SRD). To provide control for this test resource: 1. SRD and supporting documentation is reviewed		
	2. Software requirements are entered into test tool database		
	3. Draft reports are produced tracing software requirements to system development area of responsibility		
Task 2	Review Software Requirements Software Requirements are reviewed and reconciled to the SRD. To assist in Quality Assurance (QA) for this test resource: 1. Review software requirements defined to the test tool		
	2. Modifications are made to the test tool database		
	3. Software requirements are placed under configuration management (CM)		
Step 2	Publish Approved Software Requirements		
Task 1	Produce Final Reports 1. Publish report(s) that trace software requirements to system development area of responsibility		
Task 2	Submit software requirements to CM 1. Software requirements are placed under CM and change is restricted		

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 3	**Test Requirements Tracking**		
Step 1	Controlling Test Requirements Resources		
Task 1	Enter Test Requirements Test requirements are a product of the System Test Description (STD). To provide control to this test resource: 1. STD and supporting documentation is reviewed		
	2. Test requirements are entered into test tool database		
	3. Draft reports that trace test requirements to system development area of responsibility and/or software requirements		
Task 2	Review Test Requirements Test requirements are reviewed and reconciled to the STD. To assist in QA for this test resource: 1. Review test requirements as defined to the test tool		
	2. Modifications are made to test tool database		
Step 2	Publish Approved Test Requirements		
Task 1	Produce Final Reports 1. Publish report(s) that trace test requirements to system development area of responsibility		
Task 2	Submit Software Requirements to CM 1. Test requirements are placed under CM and change is restricted		

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 4	**Creating Automated Test Scripts**		
Task 1	Review STD To prepare for automated test script development: 1. STD and supporting documentation is reviewed to determine processing steps		
	2. Interfaces and pre-existing conditions identified in STD are verified to exist within test environment		
	3. Verification points (check points) identified in STD are reviewed to determine Pass/Fail criteria		
Task 2	Record or Program Test Scripts To create test scripts: 1. Place the application to be tested in the condition required to begin testing, as defined in STD		
	2. Record the steps defined in STD (alternatively write instructions to perform these steps)		
	3. Review script to ensure the system is: a) returned to the same condition as at the beginning of the step or b) in the condition necessary for the next successive script		
	4. Add verification or check points		
	5. Execute the test script to ensure correct verification results		
	6. Examine test log file to ensure that proper Pass/Fail indication has been recorded		
Task 3	Submit Test Script to CM 1. Test scripts are moved to a controlled repository from which they will be executed		

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 5	**Executing Automated Test Scripts**		
Step 1	Executing Tests in Controlled Environment		
Task 1	Review STD To prepare for test script development: 1. STD and supporting documentation is reviewed to confirm processing steps		
	2. Interfaces and pre-existing conditions identified in the STD are verified to exist within test environment		
	3. Verification points (check points) are reviewed to determine Pass/Fail criteria		
Task 2	Execute Test Scripts To execute test scripts: 1. Place the application to be tested in the condition required to begin testing, as defined in the STD		
	2. Execute the test script		
	3. Preserve the test log file and any output data files		
Step 2	Test Evaluation		
Task 1	Review for false negatives For tests in which log file indicates failure: 1. Compare actual results to expected results		
	2. Determine if a requirement has changed after the script or STD was created		
	3. Determine if an error in test environment existed		
	4. Determine if error is repeatable		
	5. Schedule test update and retest if necessary		
Task 2	Review for false positives For tests in which log file indicates passed: 1. Review high-risk areas from System Test Plan (STP), compare actual results to expected results		
	2. Review tests serving to validate fixes of previously reported SPRs, compare actual results to expected results		

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 5	**Executing Automated Test Scripts**		
Step 3	Submit Results to CM and QA		
Task 1	Record Failures in Test Log 1. Status of test is updated to indicate Tested/Failed		

Step/Task Number	Description	Responsible Party or Parties	Completion/ Duration
Part 6	**Tracking SPRs**		
Step 1	Submit SPR		
Task 1	Enter the SPR into tracking database		
Step 2	Participate in SPR Review		
Task 1	Review outstanding SPR 1. Prior to formal review, gather documentation and prepare to defend evaluation of SPR		
Task 2	Participates in SPR Review 1. Project management, test team management, developers, QA, and CM determines the disposition of submitted SPRs		
Step 3	Update the Status of System Problem Report		
Task 1	Update SPR database for new SPR 1. Disposition of SPR can be a) Opened, b) Postponed, c) Duplicate, or d) Closed by review team		
Task 2	Update SPR database for resolved SPR 1. Disposition of SPR is shown as Resolved when the development team provides an updated Software Unit (SU) to CM to be included in a re-test		
Task 3	Update SPR database for Fixed and Validated SPR 1. Once a re-test has been performed the disposition is shown as Closed		

This page intentionally left blank

APPENDIX B

SPR TRACKING DATABASE TABLE DEFINITIONS

This page intentionally left blank

Sample SPR Tracking Table Layout

Field Name	Description
Unique Identifier	An identifying field used to control each SPR as a unique entity within the Database.
Description	A narrative field used to describe the problem.
Test Case	The test case where the problem was discovered.
Software Version	The identifier used to describe the software version being tested.
Date Reported	Date field containing the date on which the SPR form was completed.
Reported By	The test engineer or observer that discovered and reported the problem.
Severity	Field containing one of the following risk factors: High, Medium, or Low.
Priority	Field containing one of the following development priorities: 1, 2, 3, 4, or 5.
Current State	One of the following status that is assigned to the SPR: Submitted, Assigned, Opened, Resolved, Closed, Postponed, or Duplicate.
Date of Current State	Date field containing the date on which the SPR was placed in its current state.
Date of Last Review	Date field containing the date on which the SPR was formally reviewed.
Assigned To	Member of the development team assigned responsibility of analysis and/or resolution.
Date Opened	Date field containing the date on which the developer begins analysis or schedules analysis and repair.
Date Resolved	Date field containing the date on which the developer has considered the problem resolved.
Date Resolution Tested	Date field containing the date on which the test engineer has attempted to validate resolution.
Date Re-opened	Date field containing the date on which the test engineer returned the SPR to the opened state because it failed validation.
Functional Area	Name of CSCI or description of the area of the system in which the problem was discovered.

This page intentionally left blank

APPENDIX C

SOFTWARE REQUIREMENT TRACKING TABLE LAYOUT

This page intentionally left blank

Sample Software Requirement Tracking Table Layout

Field Name	Description
Type	SR for software requirement.
Unique Identifier	An identifying field used to control each requirement as a unique entity within the database.
Description	A narrative field used to describe the requirement.
Author	The individual responsible for writing the requirement.
Functional Area	Name CSCI or description of the area of the system in which the requirement will be met.
Date Entered	The date on which the requirement was entered into the table.
Date Modified	The date on which the requirement was modified (if applicable).

Sample Test Requirement Tracking Table Layout

Field Name	Description
Type	TR for test requirement.
Unique Identifier	An identifying field used to control each requirement as a unique entity within the database.
Description	A narrative field used to describe the requirement.
Author	The individual responsible for writing the requirement.
Functional Area	Name CSCI or description of the area of the system in which the requirement will be met.
Test Case	The test case in which the requirement will be satisfied.
Date Entered	The date on which the requirement was entered into the table.
Date Modified	The date on which the requirement was modified (if applicable).

Sample Requirement Traceability Table Layout

Field Name	Description
Unique Test Requirement Identifier	The identifying field used in the test requirements tracking table.
Unique Software Requirement Identifier	The identifying field used in the software requirements tracking table.
Author	The individual responsible for creating the trace.
Date Traced	The date on which the trace was entered into the table.

NOTE: *This table is based on a parent child relationship in which the traceability table is a child of the test requirement table. There is a one-to-many relationship in that many software requirements can be traced to a single test requirement.*

Sample Automated Test Script Traceability Table Layout

Field Name	Description
Unique Test Requirement Identifier	The identifying field used in the test requirements tracking table.
Unique Automated Test Script Identifier	The unique name given to the automated test script in the repository.
Description	A narrative field used to describe the automated test script.
Log File Name	The unique name given to the automated test log file in the repository.
Author	The individual responsible for creating the trace.
Date Traced	The date on which the trace was entered into the table.

NOTE: *This table is based on a parent child relationship in which the traceability table is a child of the test requirement table. There is a one-to-many relationship in that many automated test scripts can be traced to a single test requirement.*